Machine Learning
Assignment 3.2

1 Introduction

The purpose of this laboratory session is to learn how to use an SVM library. The chosen one is LIBSVM.

The library can be used through Matlab and it should be included in the Matlab path. To make sure it works, just type

```
svmtrain
```

or

```
svmpredict
```

Matlab must promt the help for both functions. Typing the first command prompts the following:

```
>> svmtrain
Usage: model = svmtrain(training_label_vector, training_instance_matrix, 'libsvm_options');
libsvm_options:
-s svm_type : set type of SVM (default 0)
0 -- C-SVC
1 -- nu-SVC
2 -- one-class SVM
3 -- epsilon-SVR
4 -- nu-SVR
5 -- SVDD
6 -- R^2: L1SVM
7 -- R^2: L2SVM
-t kernel_type : set type of kernel function (default 2)
0 -- linear: u'*v
1 -- polynomial: (gamma*u'*v + coef0)^degree
2 -- radial basis function: exp(-gamma*|u-v|^2)
3 -- sigmoid: tanh(gamma*u'*v + coef0)
4 -- precomputed kernel (kernel values in training_instance_matrix)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
        : in precomputed kernels, gamma for the e-hubber cost function (default 0)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking : whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates : whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
```

```
-wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n : n-fold cross validation mode
-z corrcoeff : in cross validation mode, returns correlation coefficient if != 0 (default 0)
-j no bias : if != 0 solves unbiased (bias = 0) SVC and SVR (default 0)
```

Typing the second one produces the following output:

```
>> svmpredict
Usage: [predicted_label, accuracy, decision_values/prob_estimates] = svmpredict(testing_label_vector, testing_instance_matrix, model, 'libsvm_options')
Parameters:
  model: SVM model structure from svmtrain.
  libsvm_options:
    -b probability_estimates: whether to predict probability estimates, 0 or 1 (default 0); one-class SVM not supported yet
Returns:
  predicted_label: SVM prediction output vector.
  accuracy: a vector with accuracy, mean squared error, squared correlation coefficient.
  prob_estimates: If selected, probability estimate vector.
  Based on version 3.1. Chih-Chung Chang and Chih-Jen Lin.
```

2 SVMTRAIN inputs and outputs

2.1 Training matrix and training labels

Data vectors are defined as row vectors. Then, the input matrix must be $N \times D$, where N is the number of training vectors (or patterns or instances) and D is the dimension of the space.

The labels corresponding to the training vectors are defined as a column vector of scalars.

A simplistic example illustrates the data structure. We can generate a set of data by generating 4 Gaussians around 4 different centroids. Two are labelled with $+1$ and two with -1, $+$ and o respectively in the figure generated by the following code:

```
c=[1,1;2,1.5;2,1;3,1.5];
N=10;
X=[];
sigma=0.2;
for i=1:4
    X=[X;sigma*randn(N,2)+repmat(c(i,:),N,1)];
end
Y=[ones(1,2*N) -ones(1,2*N)];
plot(X(1:end/2,1),X(1:end/2,2),'+')
hold all
plot(X(end/2+1:end,1),X(end/2+1:end,2),'o')
hold off
```

We will use this code later.

2.2 SVM options

The SVM function needs to know what is the kind of SVM that we want to run. To this end, the user must construct a string with the options. The main ones are:

> -s svm_type : set type of SVM (default 0)
>
> -t kernel_type : set type of kernel function (default 2)
>
> -c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)

An example can be run with the previous data and the Matlab line

```
model=svmtrain(Y,X,‘-s 0 -t 0 -c 100’)
```

This sentence means that we want to train an SVM using the previous data and that the SVM must be a linear (-t 0) classifier (-s 0), and with $C = 100$.

2.3 SVM model

This structured variable is the SVM training output and contains all the parameters of the trained machine. For the example above, the output must be similar to the following (please note that the variable names may change from version to version):

```
model =

    Parameters: [5x1 double]
      nr_class: 2
       totalSV: 22
           rho: -1.1313
           obj: -15.5253
         Label: [2x1 double]
         ProbA: []
         ProbB: []
           nSV: [2x1 double]
       sv_coef: [22x1 double]
           SVs: [22x2 double]
           idx: [22x1 double]
```

```
        radius: []
```

The important ones for us are:

rho: the SVM bias (usually called b).

sv_coef: The dual coefficients (usually called α_i).

SVs: The patterns corresponding to the support vectors.

idx: the indexes of the support vectors.

The model is introduced in the test function (svmpredict). In that function, if the test labels are known, it outputs the predicted labels plus an estimate of the probability of error. If the labels are not known, the corresponding variable must be a column of arbitrary numbers (we can just use zeros).

3 Experiments

3.1 Construction of a classifier with the model parameters

Using the parameters of the training, construct a machine that classifies the training data. Compare your results with the ones of 'svmpredict'. In order to find the equation of the classifier, you must compute the primal parameter $\mathbf{w}$ from the dual ones α_i and the data.

3.2 Graphical representation of an SVM

Use the example of subsection 2.1 to plot the separating line, the two margin lines and the support vectors resulting of the SVM training. Do it for a smaller value and a higher value of parameter 'sigma' of the data generating code. Comment the results.

3.3 Estimating the structural risk

.

The objective of this exercise is to draw a plot of the actual risk ($R(\alpha)$) and the empirical risk ($R_{emp}(\alpha)$). Consider a space of 10 dimensions and a set of data that can be represented and classified in a subspace of only 3 dimensions. Thus, a linear classifier of $h = 4$ should be sufficient to properly solve the classification problem.

The following code produces a set of data:

```
function [X,Y]=data(N,sigma)
w=ones(1,10)/sqrt(10);
w1=w.*[ 1  1  1  1  1 -1 -1 -1 -1 -1];
w2=w.*[-1 -1  0  1  1 -1 -1 0  1  1];
w2=w2/norm(w2);
x(1,:)=zeros(1,10);
x(2,:)=x(1,:)+sigma*w1;
x(3,:)=x(1,:)+sigma*w2;
x(4,:)=x(3,:)+sigma*w1;
X1=x+sigma*repmat(w,4,1)/2;
X2=x-sigma*repmat(w,4,1)/2;
X1=repmat(X1,2*N,1);
X2=repmat(X2,2*N,1);
X=[X1;X2];
Y=[ones(4*2*N,1);-ones(4*2*N,1)];
Z=randperm(8*2*N);
Z=Z(1:N);
X=X(Z,:)+0.2*sigma*randn(size(X(Z,:)));
Y=Y(Z);
```

It first generates eight points describing a cube of dimension σ. Four of the points are at one side of the plane described by the equation $\Pi \equiv \mathbf{w}^\top \mathbf{x} + b$ where $\mathbf{w}_i = 1/\sqrt{10}$ and $b = 0$, at a distance $\sigma/2$, and they are labelled with -1. The other four are at the other side of the plane and they are labelled as $+1$.

Each pattern is generated by choosing at random one of the eight points and then adding a 10 dimensional Gaussian noise of standard deviation 0.2σ

It is straightforward to prove that hyperplane Π is the optimal separating hyperplane in a Bayesian sense, this is, it will produce the minimum possible actual risk.

Work out the following:

- Using the function described above, compute and represent estimates of the empirical risk and the actual risk with the following steps:

  1. Set a given value of C (between $10^{-1.5}$ and 10 should be enough).
  2. Generate a set of 100 data, which will have a standard deviation $\sigma = 1$.
  3. Train an SVM and compute its empirical error.
  4. Generate another set of data.
  5. Test the SVM and compute the test error.

  Use the previous procedure to draw a graph (in logarithmic scale) that represents:

  – The average training and test errors,

  – the difference between the average test and training errors

  for 100 values of C ranging from $10^{-1.5}$ to 10 (better results are obtained when C is logarithmically spaced.

- Comment and give an explanation of the results with respect to the concepts of actual, structural and empirical risks. In particular, give an explanation of what is an estimation of the structural risk and why the minimum actual risk does not coincide with the intersection of the other two errors.

- Draw a graph showing the actual and the empirical risks as a function of the number of data from N=10 to N=500 for the data of the previous experiment with $C = 1$. Explain the graph. Does the learning procedure seem consistent?